

Homework 2: Discovery of Frequent Itemsets and Association Rules

Hongyi Luo (hluo@kth.se)

Yage Hao (yage@kth.se)

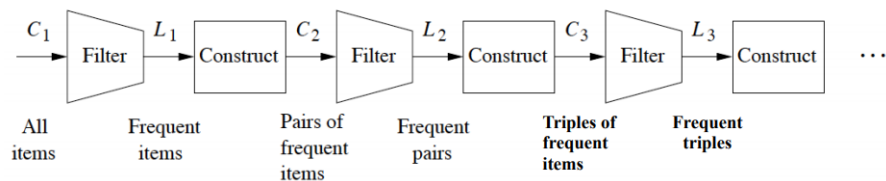
1. A Short Description of the Program

Our program is based on python, using Jupiter notebook as an interface. In the first part, we implement A-priori algorithm to find frequent itemsets given a support threshold. In the second part, we design an algorithm to mine association rules between frequent itemsets, given the support threshold and a confidence threshold. To test the implementation, we use the provided sale transaction dataset (<https://kth.instructure.com/courses/20000/files/3225774>).

2. Instructions on build and run

Task A: Frequent Itemsets

In this part, we first solve the occurrences of 1-freq, then choose candidates pairs among them. We use dictionary as our data structure to represent the item combination occurrences. Read the frequent itemsets, filter them by the setting threshold, finally generate it in the correct form. Below is the pipeline of the A-Priority algorithm which we implement in this part.



In detail, we implement two key functions corresponding to the above ‘Construct Ck’ step and ‘Filter Lk’ step respectively.

Getting the candidates item combinations. (Ck)

def get_ck_possible_itemset(dict_freq_items, receipts, n_plet). Get occurrences dictionary of a given dimension. (Ck) The basic idea of this part is if a set is frequent, then the subset must be frequent. So, we are gonna to calculate from the 1 element frequent item.

Filtering the candidates to get the final results

`def filter_under_threshold(dict_occurrences, thresh = 2)`. Given a dictionary of occurrences of the elements, and a threshold (minimum support threshold), return the dictionary of elements with at least support \geq threshold.

(Optional) Task B: Association Rules

The main idea in this part is to use the freq set we get previously to generate subset "Rules" and measure its confidence. So, we need a new class rules, a method to get all kinds of subset, and a method

to calculate the confidence. For the sake of running performance, we also need to take care of the duplications.

Rule Class

As a simple example for rule: $A \rightarrow B$, where A stands for "first" and B stands for "second". Because both can be int or list of int, so we should always take care of the data type.

Association rule extraction

def generate_association_rules(dict_freq_items, confidence). For a given confidence, we analysis the freq-set from the biggest ones. We use them to generate the subset in different order such like A,B,C,D \rightarrow (A,B,C \rightarrow D), (A,B \rightarrow C,D). There are two laws always apply: 1: If A,B,C- D is below or above confidence, so is A,B \rightarrow C,D. 2: $\text{conf}(A,B \rightarrow C,D) = \text{supp}(A,B,C,D) / \text{supp}(A,B)$.

3. Result

Our test dataset is a sale transaction dataset with 100000 baskets and 870 unique items in the universal itemset. Each item is represented by a positive integer.

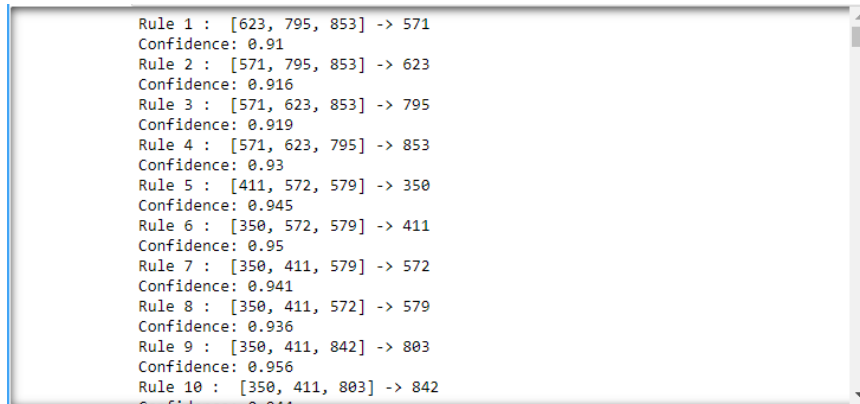
Task A: Frequent Itemsets

To test our implementation on finding frequent itemsets, we need to define argument support threshold s. We tried $s=1\%$ and $s=0.5\%$ respectively.

Support threshold	1% (typical choice)	0.5%
Number of singletons	375	569
Number of doubletons	9	342
Example itemsets in doubletons	('217', '346') ('227', '390') ('368', '682') ('368', '829') ('39', '704') ('39', '825') ('390', '722') ('704', '825') ('789', '829')	(448, 538), (39, 704), (39, 825), (704, 825), (708, 883), (708, 978), (853, 883), (883, 978), (529, 782), (227, 390),
Number of tripletons	1	110
Example itemsets in tripletons	('39', '704', '825')	(39, 704, 825), (708, 883, 978), (571, 623, 795), (571, 623, 853), (571, 795, 853), (623, 795, 853), (392, 801, 862), (350, 411, 572), (350, 411, 579), (350, 411, 803),
Number of 4-tuples	0	43

(Optional) Task B: Association Rules

We randomly picked support threshold $s=0.5\%$ and confidence threshold $c=0.1$ in this part to generate association rules. And we get 412 rules with the chosen parameters. Below is a screenshot of part of the result.



```
Rule 1 : [623, 795, 853] -> 571
Confidence: 0.91
Rule 2 : [571, 795, 853] -> 623
Confidence: 0.916
Rule 3 : [571, 623, 853] -> 795
Confidence: 0.919
Rule 4 : [571, 623, 795] -> 853
Confidence: 0.93
Rule 5 : [411, 572, 579] -> 350
Confidence: 0.945
Rule 6 : [350, 572, 579] -> 411
Confidence: 0.95
Rule 7 : [350, 411, 579] -> 572
Confidence: 0.941
Rule 8 : [350, 411, 572] -> 579
Confidence: 0.936
Rule 9 : [350, 411, 842] -> 803
Confidence: 0.956
Rule 10 : [350, 411, 803] -> 842
Confidence: 0.944
```